Booth-Encoded Karatsuba: A Novel Hardware-Efficient Multiplier

Riya JAIN[1], *Khushbu PAHWA*[2], *Neeta PANDEY*[1]

[1]Department of Electronics and Communication Engineering, Delhi Technological University, Bawana Road, Shahbad Daulatpur Village, Rohini, 110042 Delhi, India
[2]Department of Electrical Engineering, Delhi Technological University, Bawana Road, Shahbad Daulatpur Village, Rohini, 110042 Delhi, India

rj.riyajain26@gmail.com, khushbu16win@gmail.com, n66pandey@rediffmail.com

DOI: 10.15598/aeee.v19i3.4199

Article history: Received Mar 07, 2021; Revised Jun 17, 2021; Accepted Jul 14, 2021; Published Sep 30, 2021.

Abstract. *There is a recent boom being witnessed in emerging areas like IoMT (Internet of Medical Things), Artificial Intelligence for healthcare, and disaster management. These novel research frontiers are critical in terms of hardware and cannot afford to compromise accuracy or reliability. Multiplier, being one of the most heavily used components, becomes crucial in these applications. If optimized, multipliers can impact the overall performance of the system. Thus, in this paper, an attempt has been made to determine the potential of accurate multipliers while meeting minimal hardware requirements. In this paper, we propose a novel Booth-Encoded Karatsuba multiplier and provide its comparison with a Booth-Encoded Wallace tree multiplier. These architectures have been developed using two types of Booth encoding: Radix-4 and Radix-8 for 16-bit, 32-bit and 64-bit multiplications. The algorithm is designed to be parameterizable to different bit widths, thereby offering higher flexibility. The proposed mul- tiplier offers advantage of enhanced performance with significant reduction in hardware while negligibly trad- ing off the Power Delay Product (PDP). It has been observed that the performance of the proposed architecture increases with increasing multiplier size due to significant reduction in hardware and slight increase in PDP. All the architectures have been implemented in Verilog HDL using Xilinx Vivado Design Suite.*

Keywords

Accurate, Booth-encoding, Karatsuba, Wallace.

1. Introduction

Approximate multipliers [1], [2], [3], [4], [5], [6] and [7] have garnered a lot of attention from the researchers in the recent past. This is primarily due to the need to develop architectures for arithmetic computing that can reduce delay, power and hardware utilization, while relaxing the constraint on accuracy. However, there has been an upsurge in requirement to develop accurate hardware-efficient circuits for healthcare applications, resulting in emergence of novel technologies like Wireless Body Area Networks (WBAN) [8], and biomedical circuits for Brain Machine Interfaces [9]. Finite field arithmetic is another research direction that requires optimized hardware architectures, especially for coding theory and public-key cryptosystems [10], [11] and [12]. Complex field operations like exponentiation and inversion can be attained by iterative multiplication. Thus, it is necessary to design efficient multiplier. Being the core components of these systems, slight improvement in multiplication unit can offer significant performance enhancement of overall system.

Generally, multiplication operation can be bisected in two stages: (1) partial-products generation, and (2) addition of generated partial products. Research is currently underway to optimize the multiplication operation by optimizing either of the two stages. Booth encoding is being used as an effective technique to reduce the number of partial products generated in the multiplication process [13], [14] and [15] which in turn results in substantial reduction in area. Furthermore, for optimizing the second stage, Wallace tree multipliers are used as they are faster due to logarithmic depth and utilize lesser hardware by using fewest adders pos-

sible. Therefore, Booth and Wallace tree multipliers can be modified and hybridized together to obtain an optimized multiplier, namely Booth-Encoded Wallace tree multiplier. This hybrid architecture has thus been leveraged by several researchers [16], [17], [18], [19] and [20] to attain higher speed and area optimization. Hence, we have chosen the same over other architectures for comparison with our proposed multiplier.

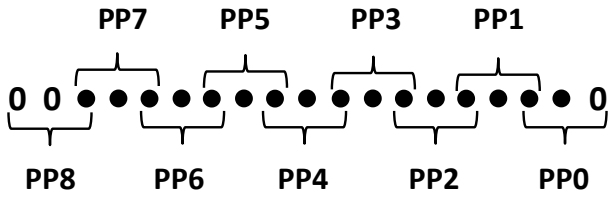

Fig. 1: Partial Products generation for 16-bit operand using Radix-4.

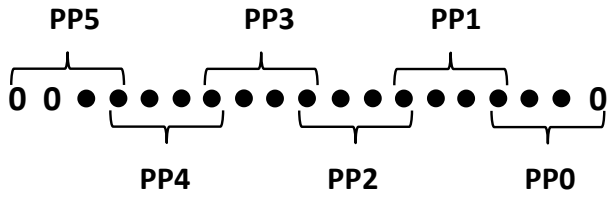

Fig. 2: Partial Products generation for 16-bit operand using Radix-8.

In Wallace tree multiplication, further optimization can be obtained by dividing an $N \times N$ multiplication in 4 small $\frac{N}{2} \times \frac{N}{2}$ multiplications which are processed in parallel [21]. The outputs of these 4 smaller size multipliers are then arranged suitably and added together using a ripple carry adder or carry look-ahead adder to obtain the final result. This helps in reducing the delay as 4 multiplications are done in parallel. However, the Karatsuba multiplier [7] uses 3 small multiplier for an $N \times N$ multiplication: two $\frac{N}{2} \times \frac{N}{2}$ multipliers and one $(\frac{N}{2}+1) \times (\frac{N}{2}+1)$ multiplier. Thus, Karatsuba multiplier requires one less multiplier compared to Wallace tree multiplier which helps in reducing the hardware drastically while causing meagre increase in delay and power due to complex structure. One less multiplier in Karatsuba algorithm is compensated by using a series of addition and subtraction operation. Therefore, the reduction in hardware becomes more significant as the size of multiplier increases. This is owed to the fact that hardware utilization increases exponentially with increase in multiplier size; however, the trend is linear for an adder. Therefore, in this paper, we have utilized the properties of Karatsuba algorithm and proposed a novel Booth-Encoded Karatsuba multiplier which is a hybrid of Booth and Karatsuba multiplier. The proposed multiplier uses one less multiplier compared to Booth-Encoded Wallace tree multiplier while individual multipliers use Booth encoding to reduce the number of partial products which is then employed with Wallace tree addition scheme.

The rest of the manuscript has been structured as follows: Sec. 2. puts forth the background of our research, focusing on Booth's algorithm, Wallace tree multiplier and how they can be hybridized to obtain an optimized Booth-Encoded Wallace tree multiplier. Section 3. elaborates on the working and implementation of proposed Booth-Encoded Karatsuba multiplier which is followed by the results in Sec. 4. Finally, the paper has been concluded in Sec. 5. based on all the findings from Sec. 4.

2. Background

2.1. Modified Booth's Algorithm

The modified Booth's algorithm [22] and [23] is a slightly modified and improved version of the Booth's algorithm. In the modified Booth algorithm, parallel encoding is performed as opposed to the serial encoding used in the originally proposed Booth algorithm. For Radix-R, encoding is obtained in following steps [24]:

- LSB of Multiplier (MR) is padded with a single bit 0.
- MR with padded 0 at LSB is partitioned into overlapping groups of x-bits where $x = 1 + \log_2 R$.
- Each block of x-bits is encoded to generate a single partial product. Encoding is done based on the Booth encoding table which differs for different Radix.
- Each partial product is then shifted s-bits to the left of its preceding neighbor, where $s = x - 1$.

For a N-bit multiplier, total of P partial products are generated, where:

$$P = \text{floor}\left(\frac{N+s}{s}\right). \tag{1}$$

All the partial products, after shifting and getting arranged suitably, are then added using appropriate addition scheme.

In this paper, Radix-4 and Radix-8 modified Booth's algorithms are being utilized. For Radix-4, partitioning of the MR is done using overlapping groups of 3-bits as shown in Fig. 1 and each block of 3-bits is encoded to generate a single partial product as per the

Tab. 1. For instance, PP0 in Fig. 1 is the the product of the encoding obtained from $\{B_{i+1},B_i,B_{i-1}\}$ and MD, where B_{i-1} is the 1'b0 appended to the right of the LSB, B_i is the 1^{st} bit (LSB), and B_i is the 2^{nd} bit. Finally, each partial product generated is shifted 2-bits left to its precedent before adding them. For an N-bit operand, the number of partial products generated is equal to floor($\frac{N+2}{2}$). Similarly, the generation of the partial products for Radix-8 is obtained by partitioning the multiplier into overlapping groups of 4-bits as shown in Fig. 2, and then each block of 4-bits is encoded as per the Tab. 2.

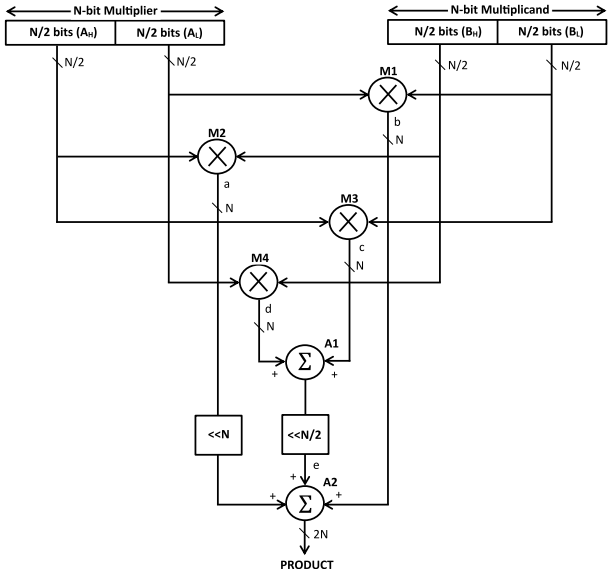

Fig. 3: Generalized Wallace Tree Architecture.[7]

Finally, before addition, all partial product is then shifted 3-bits left to its precedent. For an N-bit operand, the number of partial products generated are floor($\frac{N+3}{3}$). One factor to be noted here is that the MSB of the MR needs to be padded with at least an extra 0 bit. This is done to make the final sum a positive multiple of the Multiplicand (MD). This accounts for the appearance of 1 in the floor($\frac{N}{2}+1$) and floor($\frac{N}{3}+1$) expressions.

Thus, by means of using Booth encoding, the number of partial products generated can be reduced which in turn helps in reduction of hardware usage as well as delay.

Tab. 1: Radix-4 Booth Encoding.

B_{i+1}	B_i	B_{i-1}	Encoding
0	0	0	0× MD
0	0	1	1× MD
0	1	0	1× MD
0	1	1	2× MD
1	0	0	−2× MD
1	0	1	−1× MD
1	1	0	−1× MD
1	1	1	0× MD

Tab. 2: Radix-8 Booth Encoding.

B_{i+2}	B_{i+1}	B_i	B_{i-1}	Encoding
0	0	0	0	0× MD
0	0	0	1	1× MD
0	0	1	0	1× MD
0	0	1	1	2× MD
0	1	0	0	2× MD
0	1	0	1	3× MD
0	1	1	0	3× MD
0	1	1	1	4× MD
1	0	0	0	−4× MD
1	0	0	1	−3× MD
1	0	1	0	−3× MD
1	0	1	1	−2× MD
1	1	0	0	−2× MD
1	1	0	1	−1× MD
1	1	1	0	−1× MD
1	1	1	1	0× MD

2.2. Booth-Encoded Wallace Tree Multiplier

Several architectures have been proposed in the past with the aim of boosting the speed of the multiplier. One such architecture is the Wallace tree architecture which is an improved version of tree based multipliers [16] and [17]. The advantage offered by Wallace tree multiplier in terms of speed gets amplified when a higher multiplier units are constructed, generally 16-bit or higher. In Wallace tree architecture, the bits of all of the partial products in each column are added together by employing counters or compressors in parallel without carry propagation. A tree of Carry Save Adders (CSA) is thus used until a final 2 row matrix is generated. The final 2 rows are added using a ripple carry adder or carry look-ahead adder. The main advantage of Wallace tree architecture is the speed since the time complexity for the addition of N partial products is reduced from $O(N^2)$ to $O(\log N)$.

Furthermore, a higher order $N \times N$ bit multiplier unit can be further optimized by splitting it into smaller $\frac{N}{2} \times \frac{N}{2}$ bit multiplier units. Thus, an N-bit MD can be split into two $\frac{N}{2}$ bit words which are MD_H and MD_L. Similarly, the N-bit MR can be decomposed into MR_H and MR_L. Thus, the product for an N-bit by N-bit multiplication can be attained by employing four smaller $\frac{N}{2} \times \frac{N}{2}$ bit multiplier units. This can be better understood from Fig. 3. The products of the four multiplier units (M1, M2, M3, and M4) are given as:

$$a = MD_H \times MR_H, \tag{2}$$

$$b = MD_L \times MR_L, \tag{3}$$

$$c = MD_H \times MR_L, \tag{4}$$

$$d = MD_L \times MR_H. \tag{5}$$

Here, c and d are added together using A1 (adder) and then the sum is left shifted by $\frac{N}{2}$ bits to generate output e given as follows:

$$e = 2^{\frac{N}{2}}(c+d). \tag{6}$$

Now, a is also shifted left by N bits and added with e and b, using adder A2 to obtain the final product given by the following equation:

$$Product = 2^{N}a + 2^{\frac{N}{2}}(c+d) + b. \tag{7}$$

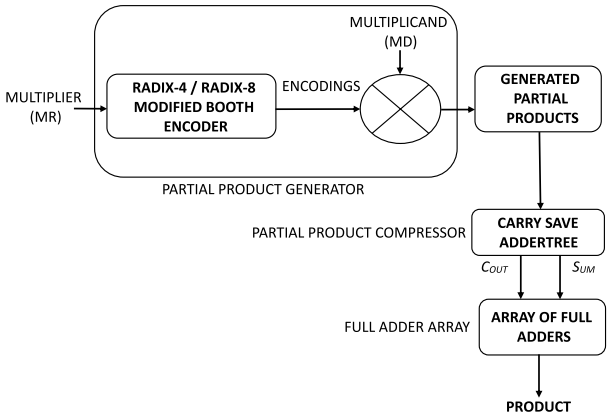

Fig. 4: Multiplier Architecture.

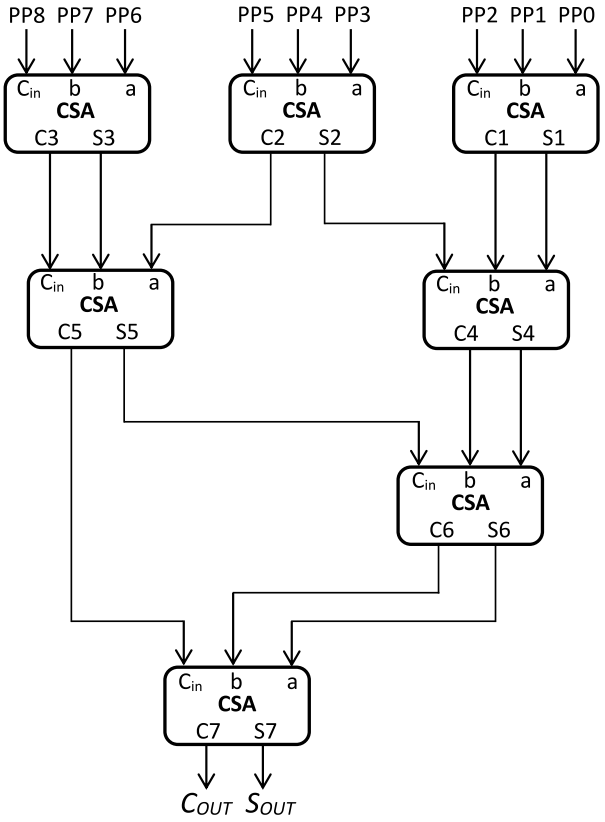

Fig. 5: CSA Tree Architecture to add 9 partial products.

Each of the 4 $\frac{N}{2}$ bit multipliers (denoted as M1, M2, M3, and M4 in Fig. 3) employ the processing steps as highlighted in Fig. 4. Various blocks involved in the multiplier architecture and their respective functionality will be explained further.

Partial Product Generator: The first processing block of the multiplier architecture, as depicted in Fig. 4, is the Partial Product Generator block that groups the MR as per the Fig. 1 or Fig. 2 depending on the Radix-4 or Radix-8 encoding. Furthermore, it generates the partial products using the encoding tables as described in Tab. 1 or Tab. 2 employing the steps enumerated in Subsec. 2.1.

Partial Product Compressor: The generated partial products are fed into the Partial Product Compressor block which is the second processing block. It employs a Wallace tree Architecture using 3:2 Carry Save Adders (CSA). For instance, the Fig. 5 represents the CSA tree architecture for the addition of 9 partial products obtained from Radix-4 Booth encoding of 16-bit multiplier. The CSA takes three inputs (a, b and cin) and generates two outputs, namely carry and sum. CSA(s) are used to improve on the delay ($O(\log N)$) and area ($O(N)$). The compression of partial products using CSA is much faster than their conventional addition. The C_{out} and S_{out} which represent the carry-out and sum generated by the last CSA block are passed onto the next block of the architecture i.e. Full Adder Array.

Full Adder Array: The Full Adder Array block takes in the C_{out} and S_{out} generated from the Partial Product Compressor block to generate the desired N-bit product by using an array of full adders, commonly known as ripple carry adder. This block can also be realized using carry look-ahead adder.

Thus, the Booth Encoded Wallace Tree Multiplier offers advantage of both Booth Encoding for reduction of Partial Products using Radix-4 and Radix-8 algorithms and faster accumulation of the generated partial products by reducing the levels of addition using the Wallace Tree structure. Thus, it finds use in various DSP applications like the computation of FFT for biomedical applications [25], and convolution operations [26].

3. Proposed Algorithm: Booth-Encoded Karatsuba Multiplier

The Karatsuba multiplication algorithm was proposed in 1962 by Karatsuba and Ofman [27]. It was the first attempt at finding efficient techniques for multiplication of large integers. The naive approach of multi-

plying two numbers has a time complexity of $O(N^2)$ for a N-bit multiplication. However, the Karatsuba algorithm leverages the divide and conquer approach and has a time complexity of $O(N^{\log_2 3})$ which is a significant improvement over the naive algorithm. The Karatsuba Algorithm can be employed in several applications that aim at achieving enhanced performance and efficient hardware accelerators, such as biomedical devices to address healthcare issues, human-computer interfaces for human assistance and cryptographic applications like RSA and DSA [28], [29] and [28]. These applications require high accuracy and demand for compact hardware designing. Recently, an approximate Karatsuba multiplier was proposed [7] in the literature which focused on the error-resilient applications. However, in this paper, we intend to introduce an accurate multiplier which integrates the Karatsuba algorithm with Booth-Encoded multiplier in order to achieve an optimized multiplier for applications where accuracy is critical and cannot be compromised.

The flow of the algorithm has been depicted in Fig. 6. In order to multiply two N-bit numbers MD and MR, the Karatsuba algorithm can be understood as described below. Let subscript H represent the upper half bits of (MD/MR) and subscript L represent the lower half bits of (MD/MR), then

$$a = MD_H \times MR_H, \tag{8}$$

$$b = MD_L \times MR_L, \tag{9}$$

At the same time,

$$c = MD_H + MD_L, \tag{10}$$

$$d = MR_H + MR_L, \tag{11}$$

Now, a and b are subtracted from the product of c and d. It can be computed as follows:

$$\begin{aligned} e &= (c \times d) - (a + b) = \\ &= (MD_H + MD_L)(MR_H + MR_L) + \\ &-(MD_H \times MR_H)(MD_L \times MR_L), \end{aligned} \tag{12}$$

After simplifying Eq. (12) we get,

$$e = MD_H MR_L + MD_L MR_H. \tag{13}$$

Thus, final product can be written as:

$$Product = 2^N a + 2^{\frac{N}{2}} e + b. \tag{14}$$

From the above equations, it can be seen that for N-bit multiplication, Karatsuba multiplier requires two $\frac{N}{2}$ bit multipliers and one $\frac{N}{2} + 1$ bit multiplier. Therefore, a total of 3 small size multipliers are required along with a series of addition and subtraction operations. In comparison to the Wallace tree multiplier as described in Subsec. 2.2. which utilises 4 small multipliers (denoted by M1, M2, M3, and M4), the Karatsuba multiplier utilises only 3 such multipliers units (M1, M2, and M3), as shown in Fig. 6. Thus, the Karatsuba multiplier performs the same N-bit multiplication as the Wallace Tree multiplier utilizing one lesser multiplier.

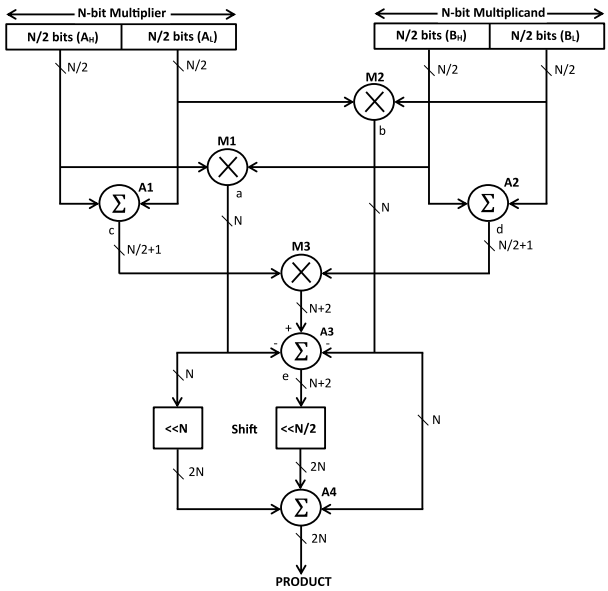

Fig. 6: Karatsuba Multiplier Architecture [7].

To further enhance the performance of the existing Karatsuba multiplier, we have proposed the hybridized architecture in which all the smaller size multipliers of Karatsuba algorithm are realized to exploit the advantages of reduced number of partial products obtained from modified Booth algorithm and the fast, parallel addition offered by Wallace Tree structure. Thus, to overcome the drawback of a conventional multiplier in which $N \times N$ multiplication results in N partial products, Booth encoding with Radix-4 and Radix-8 has been used which greatly reduces the number of partial products generated. Furthermore, these generated partial products are then added using Wallace tree addition scheme. The multiplier architecture used for BoothEncoded Karatsuba algorithm is same as described in Fig. 4. Working of multiplier architecture with respect to Booth-Encoded Karatsuba algorithm will be described below.

Partial Product Generator: The first processing block in Fig. 4 is the Partial Product Generator which uses Radix-4 or Radix-8 Booth encoding to generate partial products. For an N-bit multiplication, the proposed algorithm requires two $\frac{N}{2}$ bit multipliers and one $\frac{N}{2} + 1$ bit multiplier. In order to develop a 32-bit mul-

Tab. 3: Comparison of Total Hardware Utilization.

Size	Encoding used	Structure	Hardware	Percentage decrease
64-bit × 64-bit	Radix-4	Booth Karatsuba	2478	18.59 %
		Booth Wallace	3044	
	Radix-8	Booth Karatsuba	1815	13.57 %
		Booth Wallace	2100	
32-bit × 32-bit	Radix-4	Booth Karatsuba	777	14.05 %
		Booth Wallace	904	
	Radix-8	Booth Karatsuba	576	10.00 %
		Booth Wallace	640	
16-bit × 16-bit	Radix-4	Booth Karatsuba	252	5.97 %
		Booth Wallace	268	
	Radix-8	Booth Karatsuba	192	−14.28 %
		Booth Wallace	168	

Tab. 4: Comparison of PDP.

Size	Encoding used	Structure	Power	Delay	PDP	Percentage increase
64-bit × 64-bit	Radix-4	Booth Karatsuba	0.715	36.36	25.99	1.86 %
		Booth Wallace	0.713	35.80	25.52	
	Radix-8	Booth Karatsuba	0.719	37.40	26.89	3.73 %
		Booth Wallace	0.715	36.25	25.92	
32-bit × 32-bit	Radix-4	Booth Karatsuba	0.636	14.41	9.16	2.57 %
		Booth Wallace	0.635	14.07	8.93	
	Radix-8	Booth Karatsuba	0.639	15.23	9.73	6.45 %
		Booth Wallace	0.638	14.33	9.14	
16-bit × 16-bit	Radix-4	Booth Karatsuba	0.612	10.39	6.36	3.08 %
		Booth Wallace	0.611	10.10	6.17	
	Radix-8	Booth Karatsuba	0.612	11.52	7.05	11.46 %
		Booth Wallace	0.610	10.37	6.32	

tiplier using Radix-4 Booth encoding, two 16-bit multipliers and one 17-bit multiplier are used. For the 16-bit and 17-bit multipliers the number of partial products generated would be 9 using the Eq. (1). For the 32-bit multiplier using Radix-8 Booth encoding, the 16-bit and 17-bit multipliers would result in the generation of 6 partial products. Similarly, in order to develop a 64-bit multiplier, two 32-bit multipliers and one 33-bit multiplier are used.

Partial Product Compressor: The partial products generated in the Partial Product Generator block are then added and compressed to generate two rows of C_{out} and S_{out} using the CSA tree as described in Fig. 5.

Full Adder Array: Finally, an array of full adders is used to generate the product using C_{out} and S_{out} obtained from the Partial Product Compressor block.

4. Results

Architectures of the proposed multiplier and BoothEncoded Wallace tree multiplier [16], as discussed in Sec. 3. and Subsec. 2.2. respectively, have been realized for 16-, 32- and 64-bit multiplication using Radix-4 as well as Radix-8 Booth encoding. All the architectures have been implemented in Verilog HDL using Xilinx Vivado Design Suite. They are then evaluated in terms of hardware utilization and PDP, as discussed in this section.

Hardware Utilization: Hardware consumption for all the architectures has been compared in terms of the total number of Full Adders (FA) required. The percentage decrease in hardware of proposed architecture with respect to Booth-Encoded Wallace tree multiplier [16] have been recorded for all architectures in Tab. 3. Following observations can be derived from Tab. 3:

- For 16-, 32- and 64-bit multiplication, Radix-4 encoding, the proposed Booth-Encoded Karatsuba multiplier utilizes less hardware compared to Booth-Encoded Wallace tree multiplier.
- For 32- and 64-bit multiplication, Radix-8 encoding, the proposed Booth-Encoded Karatsuba multiplier utilizes less hardware compared to Booth Encoded Wallace tree multiplier.
- There is one outlier: for 16-bit multiplication, Radix-8 encoding, the proposed Booth-Encoded Karatsuba multiplier utilizes more hardware compared to Booth-Encoded Wallace tree multiplier. The reason for above discrepancy is that the hardware addition due to extra addition/subtraction

operations overshadows the hardware savings due to 1 less multiplier.

- Rate of decrease in hardware utilization from Booth-Encoded Wallace tree multiplier to Booth-Encoded Karatsuba multiplier increases with increase in size. Thus, system performance, in terms of hardware utilization, increases with increase in size.

- For better visualization, graphical representation has been added in Fig. 7 and Fig. 8. It can be seen that plot corresponding to Booth-Encoded Karatsuba multiplier always lies below the plot corresponding to Booth-Encoded Wallace tree multiplier - for both Radix-4 and Radix-8. Moreover, the gap between two plots keeps increasing with increasing multiplier size which shows that improvement in terms of hardware increases with increasing multiplier size.

Power Delay Product (PDP): The proposed Booth-Encoded Karatsuba architecture has also been compared with the Booth-Encoded Wallace [16] architecture in terms of the PDP. The results are summarized in the Tab. 4 from where certain observations can be made:

- For all the architectures, the proposed Booth-Encoded Karatsuba multiplier has slightly more PDP compared to Booth-Encoded Wallace tree multiplier.

- Lowest and highest PDP is seen in case of Booth-Encoded Wallace tree multiplier with Radix-4 encoding and Booth-Encoded Karatsuba multiplier with Radix-8 encoding, respectively. This trend is consistent for 16-bit, 32-bit as well as 64-bit multiplication.

- Percentage increase in PDP from Booth-Encoded Wallace tree multiplier to Booth-Encoded Karatsuba multiplier decreases with increase in size. Thus, system performance, in terms of PDP, increases with increase in size.

- For better visualization, graphical representation has been added in Fig. 9 and Fig. 10. It can be seen that plot corresponding to Booth-Encoded Karatsuba multiplier is always close to the plot corresponding to Booth-Encoded Wallace tree multiplier- for both Radix-4 and Radix-8. Thus, it is evident that there is no significant degradation in terms of PDP with varying multiplier size.

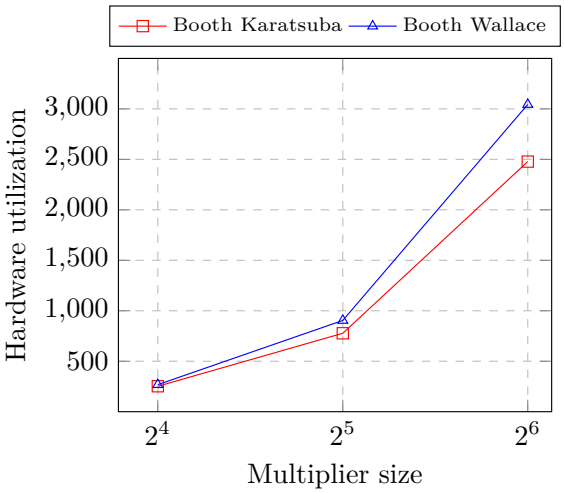

Fig. 7: Hardware Utilization varying with multiplier size for Radix-4 encoding.

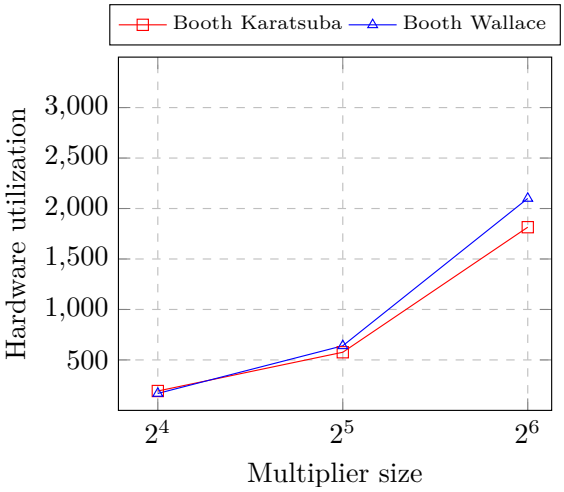

Fig. 8: Hardware Utilization varying with multiplier size for Radix-8 encoding.

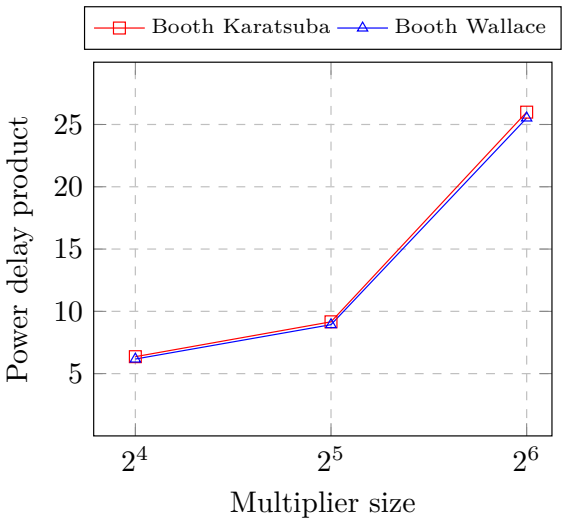

Fig. 9: PDP varying with multiplier size for Radix-4 encoding.

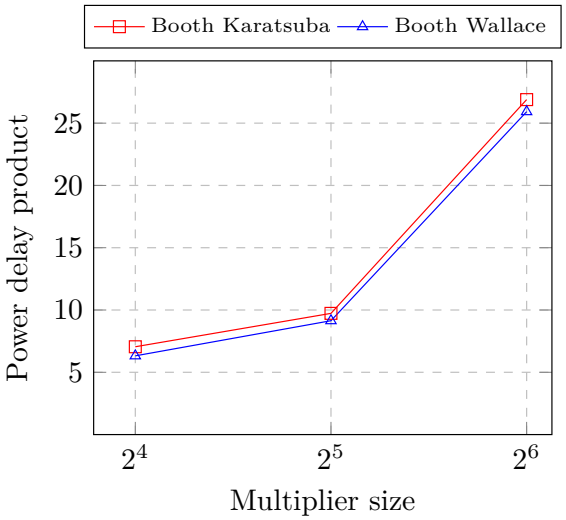

Fig. 10: PDP varying with multiplier size for Radix-8 encoding.

Hence, on the basis of above observations, we can state that there is an overall performance improvement of the proposed algorithm for 16-, 32- and 64 bit multipliers. The system's performance improvement increases with increasing size due to significant reduction in hardware utilization with slight increase in PDP. This can be accounted by the fact that percentage decrease in hardware increases with increase in multiplier size. At the same time, percentage increase in PDP decreases with increase in multiplier size. One more thing to be noted here is that when comparing different architectures with respect to encoding, Radix-4 architectures shows better performance compared to Radix-8 architectures.

5. Conclusion

An accurate multiplier based on the hybridized architecture of modified Booth encoding and Karatsuba multiplication algorithm was proposed in this paper. It was compared with an existing accurate BoothEncoded Wallace tree multiplier [16]. For 64-bit and 32-bit multiplier, there is an average improvement of 16.08 % and 12.025 % in terms of hardware while slight increase of 2.79 % and 4.51 % of PDP respectively has been encountered. Similar trend was seen for 16-bit Radix-4 multipliers. However, there is an exception seen for 16-bit Radix-8 which does not show any improvement in terms of hardware and PDP compared to the Booth-Encoded Wallace tree multiplier. Thus, on the basis of results encapsulated in previous section, it can be concluded that with increasing multiplier size, our proposed algorithm shows significant improvement. In future, the presented work can be extended by hybridizing Karatsuba multiplication method with other existing multipliers to optimize it further and leverage it in different applications. Furthermore, it can be extended to approximate computing operations to be employed in error-resilient applications.

Author Contributions

R.J. conceived the presented idea. R.J. encouraged K.P. to investigate the literature and formulate the plan for the proposed architecture. Both R.J. and K.P. developed the methodology and implemented the algorithms. R.J. performed data curation while K.P. wrote the original draft. Both R.J. and K.P. contributed to the final version of the manuscript. N.P. administered and supervised the project and contributed towards the modification of the final layout. All authors discussed the results and contributed to the final manuscript.

References

[1] GOSWAMI, S. S. P., B. PAUL, S. DUTT and G. TRIVEDI. Comparative Review of Approximate Multipliers. In: *30th International Conference Radioelektronika (RADIOELEKTRONIKA)*. Bratislava: IEEE, 2020 pp. 1–6. IBSN 978-1-7281-6469-4. DOI: 10.1109/RADIOELEKTRONIKA49387.2020.9092370.

[2] LIU, W., L. QIAN, C. WANG, H. JIANG, J. HAN and F. LOMBARDI. Design of Approximate Radix-4 Booth Multipliers for Error-Tolerant Computing. *IEEE Transactions on Computers*. 2017, vol. 66, iss. 8, pp. 1435–1441. ISSN 1557-9956. DOI: 10.1109/TC.2017.2672976.

[3] JIANG, H., C. LIU, L. LIU, F. LOMBARDI and J. HAN. A Review, Classification, and Comparative Evaluation of Approximate Arithmetic Circuits. *ACM Journal on Emerging Technologies in Computing Systems*. 2017, vol. 13, iss. 4, pp. 1–34. ISSN 1550-4832. DOI: 10.1145/3094124.

[4] MASADEH, M., O. HASAN and S. TAHAR. Comparative Study of Approximate Multipliers. In: *Proceedings of the 2018 on Great Lakes Symposium on VLSI (GLSVLSI)*. New York: ACM, 2018, pp. 415–418. ISBN 978-1-4503-5724-1. DOI: 10.1145/3194554.3194626.

[5] JIANG, H., C. LIU, N. MAHESHWARI, F. LOMBARDI and J. HAN. A comparative evaluation of approximate multipliers. In: *IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. Beijing: IEEE, 2016. ISBN 978-1-4503-4330-5. DOI: 10.1145/2950067.2950068.

[6] YAMAMOTO, T., I. TANIGUCHI, H. TOMIYAMA, S. YAMASHITA and Y. HARA-AZUMI. A systematic methodology for design and analysis of approximate array multipliers. In: *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. Jeju: IEEE, 2016 pp. 352–354. ISBN 978-1-5090-1570-2. DOI: 10.1109/APCCAS.2016.7803973.

[7] JAIN, R. and N. PANDEY. Approximate Karatsuba multiplier for error-resilient applications. *AEU - International Journal of Electronics and Communications*. 2021, vol. 130, iss. 1, pp. 1434–8411. ISSN 1434-8411. DOI: 10.1016/j.aeue.2020.153579.

[8] LIU, X., Y. ZHENG, M. W. PHYU, F. N. ENDRU, V. NAVANEETHAN and B. ZHAO. An Ultra-Low Power ECG Acquisition and Monitoring ASIC System for WBAN Applications. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*. 2012, vol. 2, iss. 1, pp. 60–70. ISSN 2156-3365. DOI: 10.1109/JETCAS.2012.2187707.

[9] CHEN, Y., E. YAO and A. BASU. A 128-Channel Extreme Learning Machine-Based Neural Decoder for Brain Machine Interfaces. *IEEE Transactions on Biomedical Circuits and Systems*. 2016, vol. 10, iss. 3, pp. 679–692. ISSN 1940-9990. DOI: 10.1109/TBCAS.2015.2483618.

[10] SHAIK, N. B. *Novel Implementation of Finite Field Multipliers over GF(2m) for Emerging Cryptographic Applications*. Dayton, 2017. Master thesis. Wright State University. Supervisor Jiafeng Xie, Ph.D.

[11] GAUBATZ, G. and B. SUNAR. Robust finite field arithmetic for fault-tolerant public-key cryptography. In: *Proceedings of the Third international conference on Fault Diagnosis and Tolerance in Cryptography (FDTC'06)*. Berlin: Springer, 2006, pp. 196–210. ISBN 978-3-540-46251-4. DOI: 10.1007/11889700_18.

[12] SAVAS, E. and C. K. KOC. Finite field arithmetic for cryptography. *IEEE Circuits and Systems Magazine*. 2010, vol. 10, iss. 2, pp. 40–56. ISSN 1558-0830. DOI: 10.1109/mcas.2010.936785.

[13] WARIS, H., C. WANG and W. LIU. Hybrid Low Radix Encoding-Based Approximate Booth Multipliers. *IEEE Transactions on Circuits and Systems II: Express Briefs*. 2020, vol. 67, iss. 12, pp. 3367–3371. ISSN 1558-3791. DOI: 10.1109/TCSII.2020.2975094.

[14] HE, Y. and C. CHANG. A New Redundant Binary Booth Encoding for Fast 2^n-Bit Multiplier Design. *IEEE Transactions on Circuits and Systems I: Regular Papers*. 2009, vol. 56, iss. 6, pp. 1192–1201. ISSN 1558-0806. DOI: 10.1109/TCSI.2008.2008503.

[15] KUANG, S.-R., J.-P. WANG and C.-Y. GUO. Modified Booth Multipliers With a Regular Partial Product Array. *IEEE Transactions on Circuits and Systems II: Express Briefs*. 2009, vol. 56, iss. 5, pp. 404–408. ISSN 1558-3791. DOI: 10.1109/TCSII.2009.2019334.

[16] ASIF, S., and Y. KONG. Performance analysis of Wallace and radix-4 Booth-Wallace multipliers. In: *Electronic System Level Synthesis Conference (ESLsyn)*. San Francisco: IEEE, 2015, pp. 17–22. ISBN 979-1-0922-7912-2.

[17] RAO, M. J. and S. DUBEY. A high speed and area efficient Booth recorded Wallace tree multiplier for fast arithmetic circuits. In: *Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics*. Hyderabad: IEEE, 2012, pp. 220–223. ISBN 978-1-4673-5067-9. DOI: 10.1109/PrimeAsia.2012.6458658.

[18] SHARMA, K., N. and S. RAVI. Modified Booth Multiplier using Wallace Structure and Efficient Carry Select Adder. *International Journal of Computer Applications*. 2013, vol. 68, no. 13, pp. 39–42. ISSN 0975-8887. DOI: 10.5120/11643-7130.

[19] SUREKA, N., R. PORSELVI and K. KUMUTHAPRIYA. An efficient high speed Wallace tree multiplier. In: *International Conference on Information Communication and Embedded Systems (ICICES)*. Chennai: IEEE, 2013, pp. 1023–1026. ISBN 978-1-4673-5788-3. DOI: 10.1109/ICICES.2013.6508192.

[20] LAKSHMANAN, T, M. OTHMAN and M. A. M. ALI. High performance parallel multiplier using Wallace-Booth algorithm. In: *Proceedings of the 9th International Conference on Neural Information Processing. Computational Intelligence for the E-Age (IEEE Cat. No.02EX575)(CONIP '02)*. Penang: IEEE, 2002, pp. 433–436. ISBN 0-7803-7578-5. DOI: 10.1109/SMELEC.2002.1217859.

[21] BHARADWAJ K., P. S. MANE and J. HENKEL. Power- and area-efficient Approximate Wallace Tree Multiplier for error-resilient systems. In: *Fifteenth International Symposium on Quality Electronic Design*. Santa Clara: IEEE, 2014, pp. 263–269. ISBN 978-1-4799-3946-6. DOI: 10.1109/ISQED.2014.6783335.

[22] MACSORLEY, O. L. High-Speed Arithmetic in Binary Computers. *Proceedings of the IRE*. 1961, vol. 49, iss. 1 pp. 67–91. ISSN 2162-6634. DOI: 10.1109/JRPROC.1961.287779.

[23] KAUR, N. and R. K. PATIAL. Implementation of Modified Booth Multiplier using Pipeline Technique on FPGA. *International Journal of Computer Applications*. 1913, vol. 68, iss. 16 pp. 38–41. ISSN 0975-8887. DOI: 10.5120/11666-7261.

[24] BEWICK, G. and M. J. FLYNN. *Binary Multiplication using Partially Redundant Multiples*. Stanford: Stanford University. 1992.

[25] AJAY, A. and M. REGEENA. VLSI Implementation of an Improved Multiplier for FFT Computation in Biomedical Applications. In: *IEEE Computer Society Annual Symposium on VLSI*. Montpellier: IEEE, 2015, pp. 68–73. ISBN 978-1-4799-8719-1. DOI: 10.1109/ISVLSI.2015.104

[26] MOHANTY, J. P., S. and R. DAS and S. K. PANDA. Design and Simulation of Convolution using Booth Encoded Wallace Tree Multiplier. *IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)*. 2016, pp. 42–46. ISSN 2278-2834.

[27] KARATSUBA, A. and Y. OFMAN. Multiplication of Multidigit Numbers on Automata. *Doklady Akadademii Nauk SSSR*. 1962, vol. 145, iss. 2, pp. 293–294.

[28] KASHIF, M., I. CICEK and M. IMRAN. A Hardware Efficient Elliptic Curve Accelerator for FPGA Based Cryptographic Applications. In: *11th International Conference on Electrical and Electronics Engineering (ELECO)*. Bursa: IEEE, 2019, pp. 362–366. ISBN 978-605-01-1275-7. DOI: 10.23919/ELECO47770.2019.8990437.

[29] CHOW, G. C. T., K. EGURO, W. LUK and P. LEONG. A Karatsuba-Based Montgomery Multiplier. In: *International Conference on Field Programmable Logic and Applications*. Milan: IEEE, 2010, pp. 434–437. ISBN 978-1-4244-7843-9. DOI: 10.1109/FPL.2010.89.

[30] MANIKANDAN, S. and C. PALANISAMY. Design of an Efficient Binary Vedic Multiplier for High Speed Applications using Vedic Mathematics with Bit Reduction Technique. *Circuits and Systems*. 2016, vol. 7, no. 9, pp. 2593–2602. ISSN 2153-1293. DOI: 10.4236/cs.2016.79224.

About Authors

Riya JAIN received her B.Tech in the discipline Electronics and Communication Engineering from Delhi Technological University (DTU) in 2020. She is a research enthusiast who aims to contribute in the field of Electronics and Communication in coming future. Her areas of interest are: Digital Very Large Scale Integration (VLSI) design, Digital Design, and Computer Architecture. She has been contributing to the field of Electronics since 2016 and has been rigorously involved in research since 2018. Being a novice researcher, she is still exploring multiple domains in the field of Electronics and is motivated enough for further contribution to technological advancement in the same.

Khushbu PAHWA received her B.Tech from Delhi Technological University (DTU) in 2020 in the field of Electrical and Electronics Engineering (Gold Medallist). Her research interests span across the broad domain of Very Large Scale Integration (VLSI) (Digital and Analog Circuits), Internet of Things (IoT), Artificial Intelligence, and Wireless Sensor Networks. She has gained valuable experience in these domains through her research internships and publications. She aspires to work on hardware optimization which can be leveraged to serve as the workhorse for future complex Deep Learning Models for deployment at the edge.

Neeta PANDEY received her M.E. in Microelectronics from Birla Institute of Technology and Sciences, Pilani in 1991 and Ph.D. from Guru Gobind Singh Indraprastha University, Delhi in 2009. She has served in Central Electronics Engineering Research Institute, Pilani, Indian Institute of Technology, Delhi, Priyadarshini College of Computer Science, Noida and Bharati Vidyapeeth's College of Engineering, Delhi in various capacities. At present, she is a professor in the ECE department, Delhi Technological University. Her teaching and research interests include analog and digital VLSI design. A life member of Indian Society for Technical Education (ISTE), and senior member of Institute of Electrical and Electronics Engineers (IEEE), USA, she has coauthored over 100 papers in international, national journals of repute and conferences.